Keith Ecker & Andy Achenreiner

2/26/2019

Exercise 3.4 Exceptions

The first vulnerability in this program was the fact that it did not catch a null pointer exception when the sql results returned null. We fixed this error by catching the null pointer exception and ignoring it.

The next vulnerability in this code was the fact that on an error too much information is displayed to the user. An attacker may be able to use this information to attack the system again or gain information or access to the system he would otherwise not have. We mitigated this by only printing just enough information for the user.

The final vulnerability of this program is that the success Boolean that is returned for the password check is defaulted to true rather than false. This causes the program to default to give access rather than default to restrict access. We mitigated this by changing the default to false and setting it to true only if the password is found in the database.

﻿

CODE ON NEXT SHEET

private static boolean checkPW(String username, String password) {

// declare database resources

Connection c = null;

Statement statement = null;

ResultSet results = null;

boolean success = false;

String sqlQuery = "SELECT COUNT(\*) AS count FROM USERS WHERE username == '" + username + "' AND password == '"

+ password + "'";

try {

// connect to the database

c = DriverManager.getConnection(DB\_URL);

// check for the username/password in database

statement = c.createStatement();

results = statement.executeQuery(sqlQuery);

// if no user with that username/password, return false

if (results.getInt("count") == 0){

success = false;

}else{

success = true;

}

} catch (SQLException ex) {

// sql error, debug info:

System.err.print("SQLQueryError\n");

}

// cleanup sql objects

try { results.close(); } catch (SQLException ex) {} catch (NullPointerException ex) { }

try { statement.close(); } catch (SQLException ex) { }

try { c.close(); } catch (SQLException ex) { }

return success;

}

}